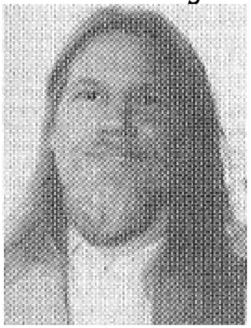
Asymmetric Cryptosystem

Prof Chik How Tan
NISlab
Gjøvik University College
Chik.tan@hig.no

In this note, some of the diagrams and materials are taken from Internet

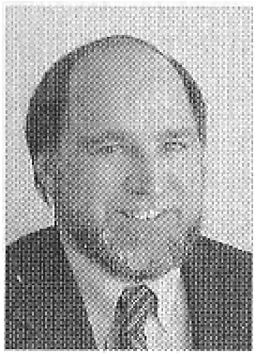
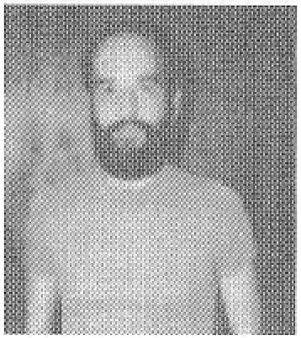
Public Key Cryptosystem

- 1976, invented by Diffie and Hellman
- 1973, also invented by Cocks, the British cryptographer. It is only release in December 1997 by British government's Communications Services Electronics Security Group (CESG)
- Main applications are the digital signature and secret key establishment over public communications channels
- This is a two keys system, that is, public key and private key



RSA Public Key Cryptosystem

- 1978, invented by R. L. Rivest, A. Shamir and L. Adleman
- This is a first to realize the public key encryption
- This cryptosystem is based on the difficulty of factorization of large number


RSA

1. Key generation
2. Encryption/Decryption
3. Digital signature generation/verification

RSA: Key Generation

1. Choose two distinct prime numbers p and q randomly.
2. Compute the product $n = p \cdot q$ and $\Phi(n)=(p-1)(q-1)$.
3. Choose an integer e randomly such that $0<e<\Phi(n)$ and $\gcd(e, \Phi(n))=1$.
4. Compute d such that $0<d<\Phi(n)$ and $e \cdot d = 1 \pmod{\Phi(n)}$.
5. Publish (n, e), keep (p, q, d) secret.

Note: e – public key (or encryption key) of Alice
d - private key (or decryption key) of Alice

RSA

Encryption : $c = E(m, e) = m^e \bmod n$

Decryption : $D(c, d) = c^d = m \bmod n$

Signature Generation : $\sigma = H(m)^d \bmod n$,

where H is a hash function

Signature Verification : $\sigma^e = H(m) \bmod n$

Security of RSA

- Factoring of n is hard
- Knowing d or $\Phi(n)$, n can be factor easily
- Share modulo n with different e_1 and e_2
- Discrete logarithm problem is also hard, that is, given m and c to find d such that $m = c^d \bmod n$

Factorization of Number

Year	Number of digits
1964	20 (~64bits)
1974	45 (~128bits)
1984	71 (~256bits)
1994	129 (~384bits)
1999	155 (~512bits)

Factoring RSA-129 (1)

This challenge was made in public in 1977 and offered a $100 to anyone who could decipher the message before 1 April, 1982.

e=9007

$$n = 114381625757888867669235779976146612010218296721242362 \\ 562561842935706935245733897830597123563958705058989075 \\ 147599290026879543541$$

The ciphertext is

$$c = 968696137546220614771409222543558829057599911245743198 \\ 746951209308162982251457083569314766228839896280133919 \\ 90551829945157815154.$$

Find the plaintext?

Factoring RSA-129 (2)

- 1994, Atkin, Graff, Lenstra and Leyland succeded in factoring RSA-129
- Involved six hundred people, with a total 1600 computers working in spare time and store the result in a large matrix
- After 7 months, a matrix with 524339 columns and 569466 rows. This matrix is spare and by Gaussian elimination reduced to the matrix with 188160 columns and 188614 rows which took 12 hours.
- After 45 hours of computation, it found the factorization of RSA-129.

Factoring RSA-129 (3)

$p=$
$3490529510847650949147849619903898133417764638493387843990820577,$

$q=$
$32769132993266709549961988190834461413177642967992942539798288533.$

$d=$
$106698614368578024442868771328920154780709906633937862801226224496631063125911774470873340168597462306553968544513277109053606095.$

Plaintext is

$200805001301070903002315180419000118050019172105011309190800151919090618010705,$

Plaintext is : the magic words are squeamish ossifrage

Factoring RSA-155 (1)

This is one of the challenge of RSA

$$\mathrm{RSA{-}155} =$$

$$109417386415705274218097073220403576120037329454492059909138421314763499842889\backslash$$

$$34784717997257891267332497625752899781833797076537244027146743531593354333897$$

Find the factor of RSA-155?

$$p =$$

$$102639592829741105772054196573991675900716567808038066803341933521790711307779$$

$$q =$$

$$106603488380168454820927220360012878679207958575989291522270608237193062808643.$$

p and q are 78 digits.

Factoring RSA-155 (2)

- August, 1999, Cavallar, Dodson, Lenstra and Lioen, Mogntgomery, Murphy, Tiele, Aradal, Gilchrist, Guillerm, Leyland, Marchand, Morain, Muffett, Putnam, Zimmermann, succeded in factoring 155 digits (512 bits)
- Initiate state take 3.7 month, on 160 SGI and Sun workstation, eight R10000 processors, 120 Pentium II PC and four DEC computer (500MHz). Total CPU time is 35.7 CPU years.
- A matrix with 6,711,336 columns and 6,699,191 rows. Finding dependencies of this matrix by Lanczos algorithm on Cray C916 took 224 hours.
- After 61.6 hours on three SGI Origin 2000 computer, it found the factorization of 155 digits.

Factoring n for given Φ(n)

We have

$$\Phi = (p-1)(q-1) = N - (p+q) + 1.$$

Hence, if we set $S = N + 1 - \Phi$, we obtain

$$S = p + q.$$

So we need to determine p and q from their sum S and product N. Define the polynomial

$$f(X) = (X-p)\cdot(X-q) = X^2 - SX + N.$$

So we can find p and q by solving $f(X) = 0$ using the standard formulae for extracting the roots of a quadratic polynomial,

$$p = \frac{S + \sqrt{S^2 - 4N}}{2},$$

$$q = \frac{S - \sqrt{S^2 - 4N}}{2}.$$

Q.E.D.

Factoring n for given Φ(n) (Con't)

As an example consider the RSA public modulus $N = 18\,923$. Assume that we are given $\Phi = \phi(N) = 18\,648$. We then compute

$$S = p + q = N + 1 - \Phi = 276.$$

Using this we compute the polynomial

$$f(X) = X^2 - SX + N = X^2 - 276X + 18\,923$$

and find that its roots over the real numbers are

$$p = 149, q = 127$$

which are indeed the factors of N.

Factoring n for given d

$$ed - 1 = s(p-1)(q-1).$$

We pick an integer $x \neq 0$, this is guaranteed to satisfy

$$x^{ed-1} \equiv 1 \pmod{N}.$$

We now compute a square root y_1 of one modulo N,

$$y_1 = \sqrt{x^{ed-1}} = x^{(ed-1)/2},$$

which we can do since $ed - 1$ is known and will be even. We will then have the identity

$${y_1}^2 - 1 \equiv 0 \pmod{N},$$

which we can use to recover a factor of N via computing

$$\gcd(y_1 - 1, N).$$

Share modulo with different e_1 and e_2

$$(N, e_1) \text{ and } (N, e_2),$$

i.e. $N_1 = N_2 = N$. Eve, the external attacker, sees the messages c_1 and c_2 where

$$c_1 = m^{e_1} \pmod N,$$
$$c_2 = m^{e_2} \pmod N.$$

Eve can now compute

$$t_1 = e_1^{-1} \pmod{e_2},$$
$$t_2 = (t_1 e_1 - 1)/e_2,$$

and can recover the message m from

$$\begin{aligned} c_1^{t_1} c_2^{-t_2} &= m^{e_1 t_1} m^{-e_2 t_2} \\ &= m^{1+e_2 t_2} m^{-e_2 t_2} \\ &= m^{1+e_2 t_2 - e_2 t_2} \\ &= m^1 = m. \end{aligned}$$

Share modulo with different e_1 and e_2

As an example of this external attack, take the public keys as

$$N = N_1 = N_2 = 18\,923,\ e_1 = 11 \text{ and } e_2 = 5.$$

Now suppose Eve sees the ciphertexts

$$c_1 = 1514 \text{ and } c_2 = 8189$$

corresponding to the same plaintext m. Then Eve computes $t_1 = 1$ and $t_2 = 2$, and recovers the message

$$m = c_1^{t_1} c_2^{-t_2} = 100 \pmod{N}.$$

ElGamal Signature Scheme

- Invented by ElGamal in 1985.
- This is based on the difficult of discrete logarithm problem over prime field
- He also invented a encryption based on discrete logarithm problem
- This scheme later modified to digital signature standard

Discrete Logarithm Problem

Let p be a prime, g be a primitive element of $Z_p^*=\{1,2,..,p-1\}$ (i.e., $Z_p^*=\{1,g,g^2,\ldots,g^{p-2}\}$).

Discrete logarithm problem: Given $y \in Z_p^*$, find the integer x such that

$$y = g^x \mod p$$

Such x is called the discrete logarithm of y over base g and denoted as $x=\log_g y$.

ElGamal : Key Generation

- Choose a large prime p and let $Z_p^*=\{1,2, \ldots,p-1\}$
- Choose a primitive element g of Z_p^*
- Randomly choose x such that $1< x <p-1$ and compute

$$y = g^x \mod p$$

(x, y) is a pair of private and public key.

Note: (p, g) may be chosen and published by a trusted third party for common use.

ElGamal : Signature Generation

Signing a message m such that $0 < m < p-1$ with the private key x

- Randomly choose a k such that $0 < k < p-2$ and $\gcd(k, p-1)=1$.
- Compute the inverse k^{-1} of k such that

$$k^{-1} \bullet k = 1 \bmod p-1$$

- Compute

$$r = g^k \bmod p$$

$$s = k^{-1}(m - x \bullet r) \bmod p-1$$

- Digital signature on m is (r,s).

ElGamal : Signature Verification

Verifying the digital signature (r,s) on the message m with the public key y

- Compute

$$u = r^s \bullet y^r \bmod p$$
$$v = g^m \bmod p$$

- Check whether $u=v$ or not. If $u=v$, then (r,s) is genuine digital signature on m. Otherwise, it is invalid.

ElGamal : Verification Equation

Prove that $r^s \bullet y^r = g^m \mod p$

Proof:

As $r = g^k \mod p$, $s = k^{-1}(m - x \bullet r) \mod p-1$

Then,

$s \bullet k = (m - x \bullet r) \mod p-1$

$= (m - x \bullet r) + i \bullet (p-1)$

$s \bullet k + x \bullet r = m + i \bullet (p-1)$,

We have

$g^{s \bullet k + x \bullet r} \mod p = g^{m + i \bullet (p-1)} \mod p$

$(g^k)^s (g^x)^r \mod p = g^m (g^{p-1})^i \mod p$

$r^s \bullet y^r \mod p = g^m \mod p$

(by Fermat Theorem: $g^{p-1} = 1 \mod p$)

Security of Signature Scheme

- *Existential forgery* : An adversary is able to forge the signature of at least one message, not necessarily the one of his/her choice
- *Selective forgery* : An adversary succeeds in forging the signature of some message of his/her choice
- *Universal forgery* : An adversary is able to forge the signature of any message without knowing the secret key
- *Retrieval of secret key* : Adversary finds out the signer's secret key

Security of ElGamal Scheme

- Knowing (p,g,y) such that $y=g^x \mod p$, it is hard for the adversary to solve the discrete logarithm problem to get the private key x of the user.
- Knowing (p,g,y,r,s), it is hard for an adversary to obtain k from $r=g^k \mod p$ and then extract the private key x of the user from $s=k^{-1}(m-x\bullet r) \mod p-1$.

The security of ElGamal signature scheme depends on the difficulty of computing discrete logarithm over Z_p.

Existential Forgery Attack to ElGamal Scheme

Without knowing the private key x of Alice, a forger chooses u, v such that gcd(v,p-1)=1 and computes

$r = y^v g^u \mod p$

$s = -rv^{-1} \mod p-1$

$m = su \mod p-1$

Then, the forged signature on m is (s, r). It can be checked that this is a valid signature as follows:

$v_1 = y^r r^s \mod p = y^r (y^v g^u)^{-rv\wedge(-1)} \mod p = (g^u)^{-rv\wedge(-1)} \mod p$

$v_2 = g^m \mod p = g^{su} \mod p = (g^u)^{-rv\wedge(-1)} \mod p$

It is obvious that $v_1 = v_2 \mod p$

Schnorr Signature Scheme

- Invented by Schnorr in 1989
- Suitable for smart card application
- Schnorr scheme is more efficient than ElGamal scheme in term of computation
- Signature size is shorter than that of ElGamal scheme

Schnorr : Parameter set up

- Let q and p are two large prime such that q divides p-1 (normally p is of 1024 bit, q is 160 bits)
- Let g be an element of Z_p^* of order q
- Let H be a hash function : $\{0,1\}^* \rightarrow Z_q$
- Choose $x < q$ and compute $y = g^x \bmod p$
- Alice's public key is (p,q, y, H); her secret key is x

Schnorr : Signature Generation

- Let m be a message in $\{0,1\}^*$
- Alice picks a random $k < q$ and computes a signature pair (e,s) where
  - $r = g^k \bmod p$
  - $e = H(m \,||\, r)$
  - $s = k + xe \bmod q$
- The signature of m is (e,s)

Schnorr : Signature Verification

- Given a message-signature pair (m, (e,s)). Bod verify the following
  - $r' = g^s y^{-e} \bmod p$
  - $e' = H(m \| r')$
  - Check $e=e'$
- If $e=e'$ then the signature is a valid one, otherwise invalid

Schnorr Signature Scheme (Example)

- $p=607$, $q=101$, $g=601$
- Let $x=3$ as a scret key, $y=g^x \bmod p = 391$ as a public key
- Let $k=65$, then $r=g^k \bmod p=223$
- $e=H(m \,||\, r) \bmod q$. Let $e=93$
- Then, $s=k+xe \bmod q = 65 + 3 \cdot 93 \bmod 101 = 41$
- Hence, the signature is (41, 93)
- Verification: $g^{41}y^{-93}=r \bmod p$

Digital Signature Standard (DSS)

FIPS PUB 186

Digital Signature Standard

Federal Information Processing Standards Publications

U. S. Department of Commerce/N.I.S.T.

May 1994

Digital Signature Standard

- Key generation:
  - generate a large random prime p such that $2^{511} < p < 2^{1024}$
  - Choose a prime factor q of p-1 such that $2^{159} < q < 2^{160}$
  - Choose an integer h such that $1 < h < p-1$ and $g = h^{(p-1)/q} \pmod{p} > 1$
  - H is a secure hash function (SHA)
  - select a random integer x, $1 \le x \le p-2$
  - compute $y = g^x \bmod p$
  - public key: (p, g, y)
  - private key: x

DSS (Cont'd)

- Signature generation:
  - select a random integer k, $0 < k < q$
  - compute $r = (g^k \bmod p) \bmod q$
  - compute $k^{-1} \bmod q$
  - compute $s = k^{-1} (H(m) + x\, r) \bmod q$
  - the signature is the pair (r, s)

DSS (Cont'd)

- Signature verification:
  - obtain authentic public key (p, q, g, y)
  - verify that $1 \le r < q$ and $1 \le s < q$
  - compute $u = s^{-1} H(m) \bmod q$ and $v = s^{-1} r \bmod q$
  - compute $z = (g^u y^v \bmod p) \bmod q$
  - accept the signature if $z = r$

DSS : Verification Equation

Prove $(g^u \bullet y^v \bmod p) \bmod q = r$

Proof:

As $r=(g^k \bmod p) \bmod q$, $s=k^{-1}(H(M)+x\bullet r) \bmod q$

and $u=s^{-1}\bullet H(M) \bmod q$, $v=s^{-1}\bullet r \bmod q$

$k = s^{-1}(H(M)+x\bullet r)=u+v\bullet x \bmod q$

$k=u+v\bullet x + i \bullet q$

$r = (g^k \bmod p) \bmod q$

$= (g^{u+v\bullet x+i\bullet q} \bmod p) \bmod q$

$= (g^u \bullet (g^x)^v \bullet (g^q)^i \bmod p) \bmod q$

$= (g^u \bullet y^v \bmod p) \bmod q$

Security of DSS

- Knowing (p,g,y) such that $y=g^x \pmod p$, it is hard for the adversary to solve the discrete logarithm over G to get the private key x of the user.
- Knowing (p,g,y,r,s), it is hard for an adversary to obtain k from $r=(g^k \bmod p) \bmod q$ and then extract the private key x of the user from $s=k^{-1}(m+x\bullet r) \bmod q$.

The security of DSS depends on the difficulty of computing discrete logarithm over G.

Identity Based Cryptosystem

- is proposed by Shamir in 1984.
- The first ID-based signature is by Guillou and Quisquater in 1988.
- The first ID-based encryption are by Boneh and Franklin in 2001, and Cook in 2001, Sakai et al in 2000, independently.
- Idea is used user identity for encryption and signature verification.
- Does not require to have public key infrastructure.

GQ ID-based Signature

- Master key generation: choose two primes p and q, let $n=pq$, choose e and d such that $e.d=1 \bmod (p-1)(q-1)$.
  d is a secret mater key, public key is (n, e).
- Private key generation: Given an ID, compute
  $x=H(ID)^d \bmod n$, give x to the user with ID.
- Signature generation: To sign a message m, choose a random $r < n$, compute $c=H(r^e \bmod n, m)$, $s=r.x^{-c} \bmod n$
  signature is (m, c, s).
- Signature verification: Given signature (m, c, s) of ID, verify $c=H(s^e H(ID)^c \bmod n, m)$.

Elliptic Curve Cryptosystem

- Discovered independently by Koblitz and Miller in 1985.
- Miller presented at the Crypto'85 Conference.
- Security is based on the hardness of Elliptic curve discrete logarithm problem (ECDLP).
- Any protocol based on DLP can be converted to one based on ECDLP.

Elliptic Curve E over Z_p

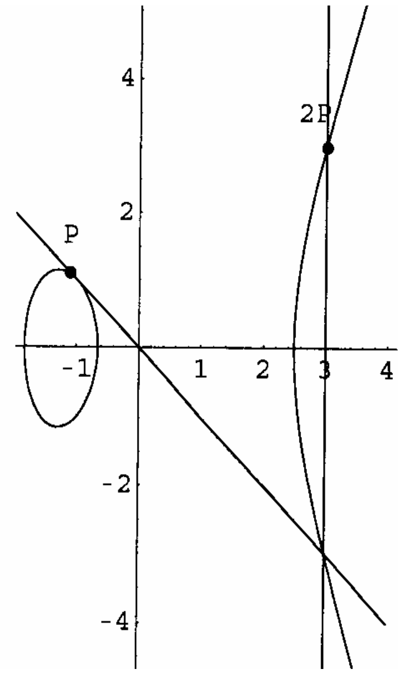
$$y^2 = x^3 + ax + b$$

Where $a, b \in Z_p$ and $4a^3 + 27b^2 \neq 0 \bmod p$

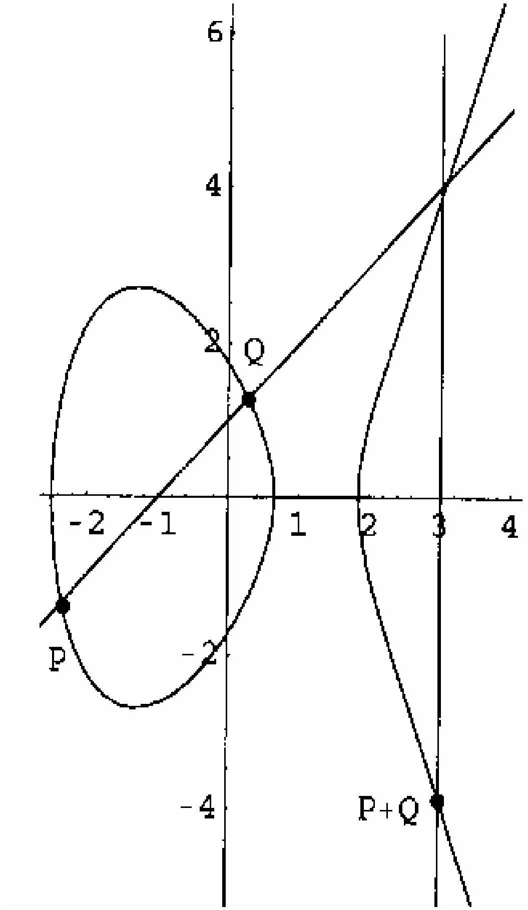
$E(Z_p)$ consists of all the point (x, y) plus a O point.

Addition of Points

- $P + O = O + P = P$ for all $P \in E(Z_p)$
- if $P=(x,y) \in E(Z_p)$, then $-P=(x,-y)$ and $(x,y) + (x,-y) = O$



Adding points on an elliptic curve



Formula for adding points

Let $P = (x_1, y_1) \in E(Z_p)$ and $Q = (x_2, y_2) \in E(Z_p)$, where $P \neq -Q$. Then $P+Q = (x_3, y_3)$, where

$$x_3 = \lambda^2 - x_1 - x_2$$

$$y_3 = \lambda(x_1 - x_3) - y_1,$$

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq Q \\ \dfrac{3x_1^2 + a}{2y_1} & \text{if } P = Q. \end{cases}$$

An Example

1. Let $P = (3, 10)$ and $Q = (9, 7)$. Then $P + Q = (x_3, y_3)$ is computed as follows:

$$\lambda = \frac{7-10}{9-3} = \frac{-3}{6} = \frac{-1}{2} = 11 \in \mathbf{Z}_{23}.$$

$$x_3 = 11^2 - 3 - 9 = 6 - 3 - 9 = -6 \equiv 17 \pmod{23}, \text{ and}$$

$$y_3 = 11(3 - (-6)) - 10 = 11(9) - 10 = 89 \equiv 20 \pmod{23}.$$

Hence $P + Q = (17, 20)$.

2. Let $P = (3,10)$. Then $2P = P + P = (x_3, y_3)$ is computed as follows:

$$\lambda = \frac{3(3^2)+1}{20} = \frac{5}{20} = \frac{1}{4} = 6 \in \mathbf{Z}_{23}.$$

$$x_3 = 6^2 - 6 = 30 \equiv 7 \pmod{23}, \text{ and}$$

$$y_3 = 6\,(3 - 7) - 10 = -24 - 10 = -11 \equiv 12 \pmod{23}.$$

Hence $2P = (7,12)$.

Elliptic Curves

Weierstraß equation:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \text{ over } \mathcal{K}$$

(i) $\mathcal{O}$ is the dentity element: $P + \mathcal{O} = P$.

(ii) The inverse of $P = (x_1, y_1)$ is $-P = (x_1, -y_1 - ax_1 - a_3)$.

(iii) If $Q = -P$, then $P + Q = \mathcal{O}$.

(iv) Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ with $Q \neq -P$. Then $P + Q =$

(x_3, y_3)

where

$$x_3 = \lambda^2 + a_1\lambda - a_2 - x_1 - x_2 \text{ and } y_3 = -(\lambda + a_1)x_3 - \mu - a_3$$

with

$$\lambda = \begin{cases} \frac{y_1 - y_2}{x_1 - x_2} & \text{if } P \neq Q \\ \frac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3} & \text{if } P = Q \end{cases}$$

and $\mu = y_1 - \lambda x_1$

Elliptic Curves over binary fields

If char($\mathcal{K}$)=2, then the elliptic curve is of the form

$$E : y^2 + xy = x^3 + ax^2 + b \text{ over } GF(2^m)$$

$$x_3 = \begin{cases} (\frac{y_1+y_2}{x_1+x_2})^2 + \frac{y_1+y_2}{x_1+x_2} + x_1 + x_2 + a & \text{if } P \neq Q \\ x_1^2 + \frac{b}{x_1^2} & \text{if } P = Q \end{cases}$$

$$y_3 = \begin{cases} (\frac{y_1+y_2}{x_1+x_2})(x_1 + x_3) + x_3 + y_1 & \text{if } P \neq Q \\ x_1^2 + (x_1 + \frac{y_1}{x_1})x_3 + x_3 & \text{if } P = Q \end{cases}$$

Correspondence between $\mathbf{Z}_p^*$ and $E(\mathrm{Z}_p)$ notation.

Group	$\mathbf{Z}_p^*$	$E(\mathrm{Z}_p)$
Group elements	Integers $\{ 1, 2, \ldots, p - 1 \}$	Points (x, y) on E plus O
Group operation	multiplication modulo p	addition of points
Notation	Elements: g, h Multiplication: $g \bullet h$ Inverse: g^{-1} Division: g / h Exponentiation: g^a	Elements: P, Q Addition: $P + Q$ Negative: $-P$ Subtraction: $P - Q$ Multiple: aP
Discrete Logarithm Problem	Given $g \in \mathbf{Z}_p^*$ and $h = g^a \bmod p$, find a	Given $P \in E(\mathrm{Z}_p)$ and $Q = aP$, find a.

Elliptic Curve Digital Signature Algorithm (ECDSA)

1999 Jan.	ANSI X9.62
2000 Jan.	FIPS 186-2
2000 Aug.	IEEE Std 1363-2000

ECDSA : Key Generation

- Domain parameters: E, $\mathbb{F}_q$, $G \in E(\mathbb{F}_q)$, $n = \text{ord}(G)$, $h = \#E(\mathbb{F}_q)/n$.
- Each entity A does the following:
  1. Select a random integer d in the interval $[1, n-1]$.
  2. Compute $Q = dG$.
  3. A's public key is Q; A's private key is d.

ECDSA : Signature Generation

To sign a message m, A does the following:

1. Select a random integer k, $1 \leq k \leq n-1$.
2. Compute $kG = (x_1, y_1)$ and $r = x_1 \bmod n$.
   If $r = 0$ then go to step 1.
3. Compute $k^{-1} \bmod n$.
4. Compute $e =$ SHA-1(m).
5. Compute $s = k^{-1}\{e + dr\} \bmod n$.
   If $s = 0$ then go to step 1.
6. A's signature for the message m is (r, s).

ECDSA : Signature Verification

To verify A's signature (r, s) on m, B should do the following:

1. Verify that r and s are integers in the interval $[1, n-1]$.
2. Compute $e = \text{SHA-1}(m)$.
3. Compute $w = s^{-1} \bmod n$.
4. Compute $u_1 = ew \bmod n$ and $u_2 = rw \bmod n$.
5. Compute $u_1G + u_2Q = (x_1, y_1)$ and $v = x_1 \bmod n$.
6. Accept the signature if and only if $v = r$.

Comparable Key Sizes

Symmetric cipher key lengths	Example algorithm	ECC key length	RSA/DL key length
80	SKIPJACK	160	1024
112	Triple-DES	224	2048
128	128-bit AES	256	3072
192	192-bit AES	384	7680
256	256-bit AES	512	15360

Computing power (Pollard rho-method)

Field size (in bits)	Size of n (in bits)	$\sqrt{\pi n / 2}$	MIPS years
163	160	2^{80}	9.6×10^{11}
191	186	2^{93}	7.9×10^{15}
239	234	2^{117}	1.6×10^{23}
359	354	2^{177}	1.5×10^{41}
431	426	2^{213}	1.0×10^{52}

Elliptic Curve Key Size (by NIST)

Symmetric cipher key length	Example algorithm	Bitlength of p in prime field $\mathbb{F}_p$	Dimension m of binary field $\mathbb{F}_{2^m}$
80	SKIPJACK	192	163
112	Triple-DES	224	233
128	AES Small [25]	256	283
192	AES Medium [25]	384	409
256	AES Large [25]	521	571

NIST-recommended elliptic curve over prime fields

P-192: $p = 2^{192} - 2^{64} - 1$, $a = -3$, $h = 1$,
$b =$ 0x 64210519 E59C80E7 0FA7E9AB 72243049 FEB8DEEC C146B9B1
$n =$ 0x FFFFFFFF FFFFFFFF FFFFFFFF 99DEF836 146BC9B1 B4D22831
P-224: $p = 2^{224} - 2^{96} + 1$, $a = -3$, $h = 1$,
$b =$ 0x B4050A85 0C04B3AB F5413256 5044B0B7 D7BFD8BA 270B3943 2355FFB4
$n =$ 0x FFFFFFFF FFFFFFFF FFFFFFFF FFFF16A2 E0B8F03E 13DD2945 5C5C2A3D
P-256: $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$, $a = -3$, $h = 1$,
$b =$ 0x 5AC635D8 AA3A93E7 B3EBBD55 769886BC 651D06B0 CC53B0F6 3BCE3C3E 27D2604B
$n =$ 0x FFFFFFFF 00000000 FFFFFFFF FFFFFFFF BCE6FAAD A7179E84 F3B9CAC2 FC632551
P-384: $p = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$, $a = -3$, $h = 1$,
$b =$ 0x B3312FA7 E23EE7E4 988E056B E3F82D19 181D9C6E FE814112 0314088F 5013875A C656398D 8A2ED19D 2A85C8ED D3EC2AEF
$n =$ 0x FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF C7634D81 F4372DDF 581A0DB2 48B0A77A ECEC196A CCC52973
P-521: $p = 2^{521} - 1$, $a = -3$, $h = 1$,
$b =$ 0x 00000051 953EB961 8E1C9A1F 929A21A0 B68540EE A2DA725B 99B315F3 B8B48991 8EF109E1 56193951 EC7E937B 1652C0BD 3BB1BF07 3573DF88 3D2C34F1 EF451FD4 6B503F00
$n =$ 0x 000001FF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFA 51868783 BF2F966B 7FCC0148 F709A5D0 3BB5C9B8 899C47AE BB6FB71E 91386409

$$y^2 = x^3 + ax + b$$

The number of points on E is nh

NIST-recommended elliptic curve over binary fields

B-163: $a = 1$, $h = 2$, $f(x) = x^{163} + x^7 + x^6 + x^3 + 1$

$b =$ 0x 00000002 0A601907 B8C953CA 1481EB10 512F7874 4A3205FD

$n =$ 0x 00000004 00000000 00000000 000292FE 77E70C12 A4234C33

B-233: $a = 1$, $h = 2$, $f(x) = x^{233} + x^{74} + 1$

$b =$ 0x 00000066 647EDE6C 332C7F8C 0923BB58 213B333B 20E9CE42 81FE115F 7D8F90AD

$n =$ 0x 00000100 00000000 00000000 00000000 0013E974 E72F8A69 22031D26 03CFE0D7

B-283: $a = 1$, $h = 2$, $f(x) = x^{283} + x^{12} + x^7 + x^5 + 1$

$b =$ 0x 027B680A C8B8596D A5A4AF8A 19A0303F CA97FD76 45309FA2 A581485A F6263E31 3B79A2F5

$n =$ 0x 03FFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFEF90 399660FC 938A9016 5B042A7C EFADB307

B-409: $a = 1$, $h = 2$, $f(x) = x^{409} + x^{87} + 1$

$b =$ 0x 021A5C2 C8EE9FEB 5C4B9A75 3B7B476B 7FD6422E F1F3DD67 4761FA99 D6AC27C8 A9A197B2 72822F6C D57A55AA 4F50AE31 7B13545F

$n =$ 0x 01000000 00000000 00000000 00000000 00000000 00000000 000001E2 AAD6A612 F33307BE 5FA47C3C 9E052F83 8164CD37 D9A21173

B-571: $a = 1$, $h = 2$, $f(x) = x^{571} + x^{10} + x^5 + x^2 + 1$

$b =$ 0x 02F40E7E 2221F295 DE297117 B7F3D62F 5C6A97FF CB8CEFF1 CD6BA8CE 4A9A18AD 84FFABBD 8EFA5933 2BE7AD67 56A66E29 4AFD185A 78FF12AA 520E4DE7 39BACA0C 7FFEFF7F 2955727A

$n =$ 0x 03FFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF E661CE18 FF559873 08059B18 6823851E C7DD9CA1 161DE93D 5174D66E 8382E9BB 2FE84E47

$$y^2 + xy = x^3 + ax^2 + b$$

The number of points on E is nh

NIST-recommended elliptic curve over binary fields

$$y^2 + xy = x^3 + ax^2 + b$$

The number of points on E is nh

K-163: $a = 1$, $b = 1$, $h = 2$, $f(x) = x^{163} + x^7 + x^6 + x^3 + 1$

$n =$ 0x 00000004 00000000 00000000 00020108 A2E0CC0D 99F8A5EF

K-233: $a = 0$, $b = 1$, $h = 4$, $f(x) = x^{233} + x^{74} + 1$

$n =$ 0x 00000080 00000000 00000000 00000000 00069D5B B915BCD4 6EFB1AD5 F173ABDF

K-283: $a = 0$, $b = 1$, $h = 4$, $f(x) = x^{283} + x^{12} + x^7 + x^5 + 1$

$n =$ 0x 01FFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFE9AE 2ED07577 265DFF7F 265DFF7F 94451E06 1E163C61

K-409: $a = 0$, $b = 1$, $h = 4$, $f(x) = x^{409} + x^{87} + 1$

$n =$ 0x 007FFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFE5F 83B2D4EA 20400EC4 557D5ED3 E3E7CA5B 4B5C83B8 E01E5FCF

K-571: $a = 0$, $b = 1$, $h = 4$, $f(x) = x^{571} + x^{10} + x^5 + x^2 + 1$

$n =$ 0x 02000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 131850E1 F19A63E4 B391A8DB 917F4138 B630D84B E5D63938 1E91DEB4 5CFE778F 637C1001

Software Timing for ECDSA

In 2000, M. Aydos, T. Tank, and C. K. Koc implemented ECDSA over Z_p in 80MHz 32-bits ARM7TDMI

ECDSA	160	176	192	208	256
Signing	46.4ms	65.4ms	71.3ms	96.2ms	153.5ms
Verifying	92.4ms	131.3ms	148.3ms	194.3ms	313.4ms

Software Timing for ECDSA (Cont'd)

Curve type	NIST Curve	Signing (ms)	Verification (ms)
Prime	P-192	0.28	0.938
	P-224	0.41	1.38
	P-256	0.686	2.25
Binary	B-163	0.48	1.47
	B-233	1.18	3.58
	B-283	1.80	5.385
Koblitz	K-163	0.385	0.79
Binary	K-233	0.842	1.73
	K-283	1.23	2.55

Timing for *k*.P on FPGA implementation

Target Platform	Key Size	$k \cdot P$ Operations per second
FPGA Hardware [12] (XCV300, 36 MHz)	155	148
FPGA Hardware [12] (XCV300, 33 MHz)	281	70
FPGA CryptoProcessor (XC4085XLA, 37 MHz)	155	775
FPGA CryptoProcessor (XC4085XLA, 36 MHz)	191	431
FPGA CryptoProcessor (XC4085XLA, 34 MHz)	270	146
ASIC CryptoProcessor (AWP, 1 GHz)	270	2300 (estimated)

Over binary field

Core ECC Standards

Standard	Schemes included
ANSI X9.62	ECDSA
ANSI X9.63	ECIES, ECDH, ECMQV
FIPS 186-2	ECDSA
IEEE P1363	ECDSA, ECDH, ECMQV
IEEE P1363A	ECIES
IPSec	ECDSA, ECDH
ISO 14888-3	ECDSA
ISO 15946	ECDSA, ECDH, ECMQV

ECDSA vs RSA (ms)

	Elliptic curve over $\mathbb{F}_{2^{233}}$		
	RIM pager	PalmPilot	Pentium II
Key Generation	1,552	2,573	3.11
ECDSA Signing	1,910	3,080	4.03
ECDSA Verifying	3,701	5,878	7.87

	2048-bit modulus		
	RIM pager	PalmPilot	Pentium II
RSA Key Generation	—	—	26,442
RSA Signing	111,956	288,236	440.69
RSA Verifying ($e = 3$)	1,087	2,392	4.2
RSA Verifying ($e = 2^{16} + 1$)	3,608	7,973	13.45

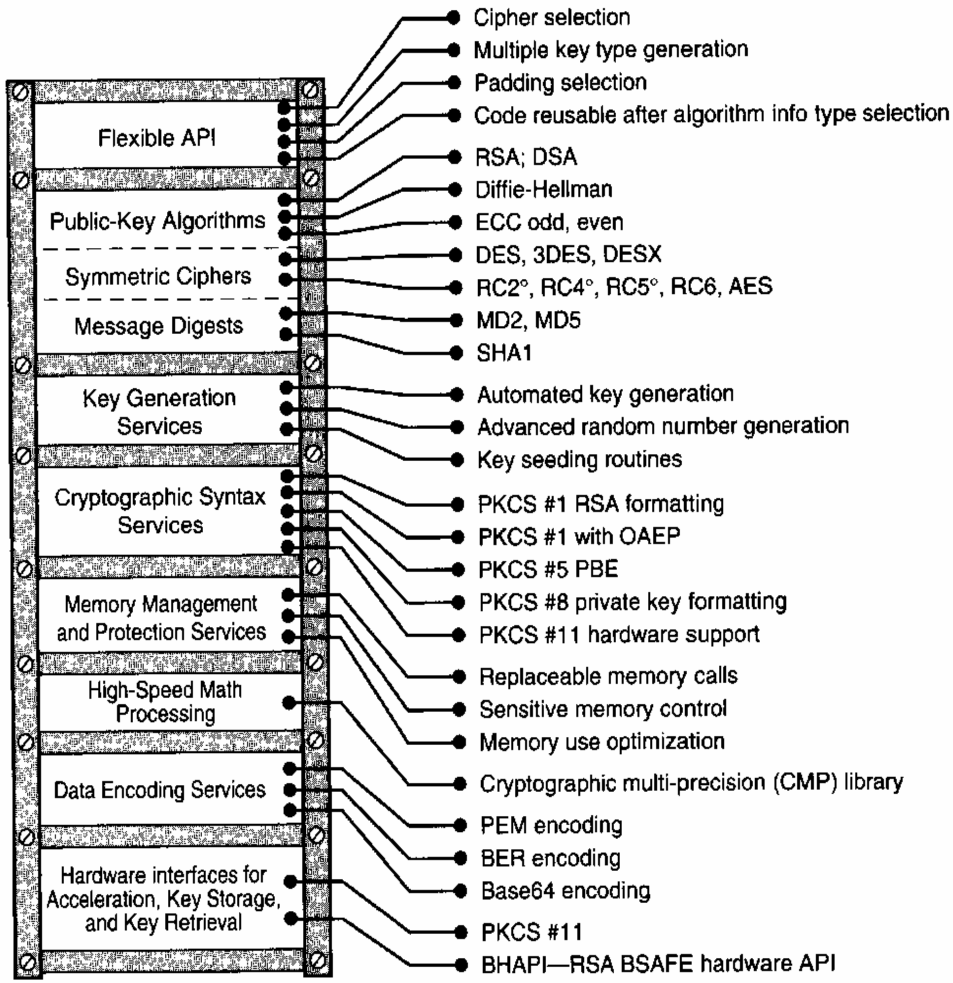
ECC – Patent Situation

- The general idea to use elliptic curve for public key cryptosystem is not patented
- All the relevant public key based security services are patent free, digital signature, key exchange, authentication
- Some elliptic curve analogues cryptographic schemes are patented, example, Menezes-Qu-Vanstone, Nyberg-Rurppel, Schnorr, etc
- There are a large number of patents on special implementation techniques.

Some Patents

- J.L Messay and J.K. Omura. Computational method and apparatus for finite field arithmetic. US Patent 4,587,627, May, 1986.
- R.C. Mullin, I.M. Onyszchuk, and S.A. Vanstone. Computational Method and apparatus for finite field multiplication, US Patent 4,745,568, May, 1988.
- R.C. Mullin. Multiple bit multiplier. US Patent 5,787,028, Jul, 1998.
- P. Ning and Y.L. Yin Efficienct software implementation for finite field multiplication in normal basis. Pending US Patent application. filed in Dec 1997.
- R.J. Lambert and A. Vadekar. Method and apparatus for finite field multiplication. US Patent 6,049,815, April 2000.
- C. K. Koc, E. Savas, and A. F. Tenca. A Scalable and Unified Multiplier for Finite Fields. US Patent Application, February, 2000.
- C. K. Koc, A. F. Tenca, and G. Todorov. An high-radix scalable modular multiplier. US Patent Application, April, 2001.

RSA BSAFE Crypto-C Functional Layers



Notions of Cryptographic Security

- Unconditional Security: There is no bound place on the amount of computation that an adversary is allowed to carry out.
- Computational Security: This measure concerns the computational effort to break a cryptosystem.
- Provable Security: Provide evidence of security by reducing the security of cryptosystem to well-studies mathematical problem that is believed to be difficult to solve. This is also refer to *reductionist security*.